Bildungszentrum Uster
Berufsfachschule Uster
Wirtschaft und Technik

# Modul 114

# Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen

**Walter Rothlin**
Bildungszentrum Uster

Letzte Änderungen: 21.9.19

# Modulidentifikation Modul 114

Geeignete Kompressionsverfahren für die Archivierung von Information auswählen und einsetzen. Dabei die erforderlichen Vorkehrungen für eine langfristige Wiederverwendung der Informationen beachten und die dazu erforderlichen Massnahmen treffen.

**Codierungen** und **Kompressionsverfahren** für die Übertragung von Informationen auswählen und Voraussetzungen definieren, die einen problemlosen Austausch codierter und komprimierter Informationen ermöglichen.

**Verschlüsselungsverfahren** zur Sicherung von Informationen gegen unbefugten Zugriff auf Speichern und Übertragungswegen auswählen und einsetzen.

Gesicherte Übertragungsverfahren für Dateien mit asymmetrischen und symmetrischen Verschlüsselungsverfahren nutzen. Dabei Aspekte wie Public/Private Key, Zertifikate, Protokolle und Standards berücksichtigen

# Lernziele

- Grundlagen Codierung
- Sie kennen den Binärcode
- Sie kennen die Begriffe Redundanz und Hammingdistanz
- Sie können einen Hamming-Code erzeugen und Fehler korrigieren

# Was versteht man unter Codierung?

- Zuordnung

Haus

Hund

Auto

- Eine Übersetzung

Haus → House

Hund → Dog

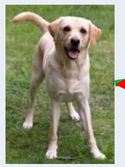
Car → Auto

- Adressierung (Eindeutigkeit), Kennzeichnen




# Eindeutigkeit - Eineindeutigkeit

Als eine (Übersetzungs-Tabelle) vorstellbar

| Objekt / Gegenstand | Code |
|---|---|
|  | Haus<br>House<br>Villa von Hans Meier in Uster |
|  | Hund<br>Dog<br>Hund mit Marke 123456789<br>Struppi |
|  | Auto<br>Car<br>Selbstfahrendes Kleinfahrzeug |
|  | Blaues Auto<br>Car with plate SZ 3620 |
| | |

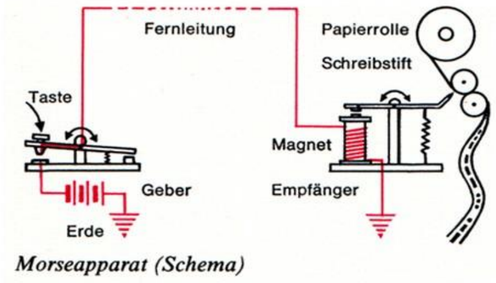
Codes sind z.B.

- Deutsche Sprache
- English
- …..

Eindeutige Zuordnung

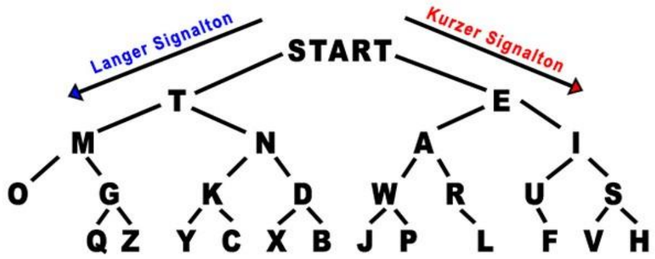
Ein-Eindeutige Zuordnung

# Beispiele von Codierung?



*Morseapparat (Schema)*

Was heisst • • • — ?



Der Morsecode als binärer Baum

| | | | | | |
|---|---|---|---|---|---|
| A | •– | J | •––– | S | ••• |
| B | –••• | K | –•– | T | – |
| C | –•–• | L | •–•• | U | ••– |
| D | –•• | M | –– | V | •••– |
| E | • | N | –• | W | •–– |
| F | ••–• | O | ––– | X | –••– |
| G | ––• | P | •––• | Y | –•–– |
| H | •••• | Q | ––•– | Z | ––•• |
| I | •• | R | •–• | | |

Der Morsecode als Codetabelle

# Beispiele von Codierung?

- **Codierungen** (Codecs)
  - MP3 (Audio)
  - JPEG (Bild)
  - MPEG2 (Film)
- **Codes**
  - **Leitungscodes** (Signalübertragung)
    - NRZ-Code
    - Manchester-Code
  - **Binäre Codes**
    - **Strich-Codes**
      - EAN-Codes
    - **Alphanum.-Codes** (Text)
      - ASCII-Code
      - Unicode
    - **Numerische Codes** (Dezimalzahlen)
      - Dual-Code
      - BCD
      - Gray-Code
      - Aiken-Code
      - Exzess-3-Code
      - 1-aus-n-Code
      - 7-Segment-Anz.

ARJ

# Definition des Begriffs «CODE»

**Definition: Eindeutige Abbildung der Zeichen des Zeichenvorrats 1 auf die Zeichen des Zeichenvorrats 2**

Vorschrift, wie Nachrichten oder Befehle zur Übertragung oder Weiterverarbeitung für ein Zielsystem umgewandelt werden. Beispielsweise stellt der Morsecode eine Beziehung zwischen Buchstaben und einer Abfolge kurzer und langer Tonsignale her

# Welche Anforderungen können an einen Code gestellt werden?

- **Ökonomische Darstellung** (Übertragungsgeschwindigkeit, Speicherplatzbedarf)
- **Sicherung gegen Verfälschung** (Übertragungsfehler, Verarbeitungsfehler)
- **Schutz vor unbefugtem Zugriff** (Verschlüsselung/Kryptologie)

# Datenverarbeitung

Computer hat die Aufgabe, eingegebene Daten zu verarbeiten

- Texte, Zahlen, Bilder, etc.
- Computer kann lediglich zwei Zustände unterscheiden
- Entweder es fliesst Strom, oder es fliesst keiner (es hat eine Spannung, oder nicht)
- Daher bedient man sich des einfachsten Zahlensystems, des Binär- oder Dualsystems

...

# Binärcode

Einfachstes Zahlensystem

- nur zwei mögliche Schaltungszustände
  - wahr – falsch / True - False
  - 0–1
  - ja – nein
  - High – Low usw.
- Zustand wird in einem Bit (Binary Digit) festgehalten
- 0 oder 1 (jedes binäre Zeichen nennt man 1 Bit)
- Kombination von mehreren Bits ergibt einen Code
- 8 Bits = 1 Byte

Kombinationen erzeugen

| $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| .. | .. | .. | .. |

# Hex, Oktal, Dec, 2er-Potenzen

| Binär | Octal | Dezimal | Hexadezimal |
|---|---|---|---|
| 0000 | 00 | 00 | 0 |
| 0001 | 01 | 01 | 1 |
| 0010 | 02 | 02 | 2 |
| 0011 | 03 | 03 | 3 |
| 0100 | 04 | 04 | 4 |
| 0101 | 05 | 05 | 5 |
| 0110 | 06 | 06 | 6 |
| 0111 | 07 | 07 | 7 |
| 1000 | 10 | 08 | 8 |
| 1001 | 11 | 09 | 9 |
| 1010 | 12 | 10 | A |
| 1011 | 13 | 11 | B |
| 1100 | 14 | 12 | C |
| 1101 | 15 | 13 | D |
| 1110 | 16 | 14 | E |
| 1111 | 17 | 15 | F |

Hex, Octal, Dec kürzere (weniger Stellen) Codes als Binär

| Basis | Exponent | Potenz |
|---|---|---|
| 2 | 0 | 1 |
| | 1 | 2 |
| | 2 | 4 |
| | 3 | 8 |
| | 4 | 16 |
| | 5 | 32 |
| | 6 | 64 |
| | 7 | 128 |
| | 8 | 256 |
| | 9 | 512 |
| | 10 | 1'024 |
| | 16 | 65'536 |
| | 20 | 1'048'576 |
| | 32 | 4'294'967'296 |

# Zeichenvorrat beim Binär-Code

- Berechnung möglicher Kombinationen (Zeichenvorrats)
  - Anzahl Kombinationen= $2^{\text{Stellenanzahl}}$
  - Allgemeine Form: Anzahl Kombinationen= (Anzahl Zeichen pro Stelle)$^{\text{Stellenanzahl}}$

## Berechnen Sie die Anzahl Kombinationen:

Anzahl Kombinationen:

| Bezeichnung | Zeichenvorrat / Stelle | Anzahl Zeichen / Stelle | Stellenzahl | | Anzahl Kombinationen |
|---|---|---|---|---|---|
| Dezimal | | | 3 | ==> | |
| Oktal | | | 4 | ==> | |
| Hexadezimal | | | 4 | ==> | |
| Alphabet | | | 2 | ==> | |
| Binär | | | 16 | ==> | |
| Mastermind | rot,grün,orange, blau,gelb | | 4 | ==> | |
| | Beliebig: | 16 | 2 | ==> | 256 |

# Zeichenvorrat beim Binär-Code (Lösung)

- Berechnung möglicher Kombinationen (Zeichenvorrats)
  - Anzahl Kombinationen= $2^{\text{Stellenanzahl}}$
  - Allgemeine Form: Anzahl Kombinationen= (Anzahl Zeichen pro Stelle)$^{\text{Stellenanzahl}}$

## Berechnen Sie die Anzahl Kombinationen:

Anzahl Kombinationen:

| Bezeichnung | Zeichenvorrat / Stelle | Anzahl Zeichen / Stelle | Stellenzahl | | Anzahl Kombinationen |
|---|---|---|---|---|---|
| Dezimal | 0..9 | 10 | 3 | ==> | 1'000 |
| Oktal | 0..7 | 8 | 4 | ==> | 4'096 |
| Hexadezimal | 0..9, A..F | 16 | 4 | ==> | 65'536 |
| Alphabet | A..Z | 26 | 2 | ==> | 676 |
| Binär | 0,1 | 2 | 16 | ==> | 65'536 |
| Mastermind | rot,grün,orange, blau,gelb | 5 | 4 | ==> | 625 |
| | Beliebig: | 16 | 2 | ==> | 256 |

# Umrechnen von Werten in unterschiedlichen Systemen

## Stellenwerte (Umwandlung: Beliebieges Zahlensystem ins Dezimale)

| Stelle | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Exponent | 16^8 | 16^7 | 16^6 | 16^5 | 16^4 | 16^3 | 16^2 | 16^1 | 16^0 |
| Stellenwerte | 4'294'967'296 | 268'435'456 | 16'777'216 | 1'048'576 | 65'536 | 4'096 | 256 | 16 | 1 |
| Vorgebener Wert | 0 | 0 | 0 | 0 | F | F | 7 | B | 1 |
| Wert an der Stelle | 0 | 0 | 0 | 0 | 15 | 15 | 7 | 11 | 1 |
| Wert * Stellenwert | 0 | 0 | 0 | 0 | 983'040 | 61'440 | 1'792 | 176 | 1 |

| | |
|---|---|
| Eingabe | FF7B1 |
| Basis | 16 |
| Dezimalwert | 1'046'449 |

FF7B1 im 16 System --> 1046449 im Dezimalen

1'046'449

## Fortlaufende Division (Umwandlung= Dezimalzahl in eine Zahl in beliebiges Zahlensystem)

| Zahl | Zahl Mod 16 | | Rest | |
|---|---|---|---|---|
| 3451 | 215 | | 11 | B |
| 215 | 13 | | 7 | 7 |
| 13 | 0 | | 13 | D |
| 0 | 0 | | 0 | |
| 0 | 0 | | 0 | |
| 0 | 0 | | 0 | |

| | | |
|---|---|---|
| Dezimalwert | 3'451 | |
| Gewünschte Basis | 16 | |
| Wert | D7B | D7B |

3451 im Dezimalsystem --> D7B im 16 System

# Anwendungsbeispiel: Umrechnen

**RGB-Farbcodes**

| Umrechnung #Hex in RGB | | | | | | |
|---|---|---|---|---|---|---|
| ffffff | FFFFFF | FF | F | 15 | 240 | 255 |
| | | | F | 15 | 15 | |
| | | FF | F | 15 | 240 | 255 |
| | | | F | 15 | 15 | |
| | | FF | F | 15 | 240 | 255 |
| | | | F | 15 | 15 | |
| **Umrechnung RGB in #Hex** | | | | | | |
| 30 | 1 | 1 | | | 1E | 1E7EB1 |
| | 14 | E | | | | |
| 126 | 7 | 7 | | | 7E | |
| | 14 | E | | | | |
| 177 | 11 | B | | | B1 | |
| | 1 | 1 | | | | |

# ASCII-Codierung

Was für einem Zeichen entspricht 00110011?

Welches Bit-Muster hat ein M?

Welchem Zeichen entspricht $2D_{16}$?

| | | | | Bit 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Bit 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | Bit 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | Bit 4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Bit 3 | Bit 2 | Bit 1 | Bit 0 | Hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 0 | 0 | 0 | 1 | 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0 | 0 | 1 | 0 | 2 | STX | DC2 | " | 2 | B | R | b | r |
| 0 | 0 | 1 | 1 | 3 | EXT | DC3 | # | 3 | C | S | c | s |
| 0 | 1 | 0 | 0 | 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0 | 1 | 0 | 1 | 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 0 | 1 | 1 | 0 | 6 | ACK | SYN | & | 6 | F | V | f | v |
| 0 | 1 | 1 | 1 | 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 1 | 0 | 0 | 0 | 8 | BS | CAN | ( | 8 | H | X | h | x |
| 1 | 0 | 0 | 1 | 9 | HAT | EM | ) | 9 | I | Y | i | y |
| 1 | 0 | 1 | 0 | A | IF | SUB | * | : | J | Z | j | z |
| 1 | 0 | 1 | 1 | B | VT | ESC | + | ; | K | [ | k | { |
| 1 | 1 | 0 | 0 | C | FF | FS | , | < | L | \ | l | \| |
| 1 | 1 | 0 | 1 | D | CR | GS | - | = | M | ] | m | } |
| 1 | 1 | 1 | 0 | E | SO | RS | . | > | N | ^ | n | ~ |
| 1 | 1 | 1 | 1 | F | SI | US | / | ? | O | _ | o | DEL |

https://www.electronicdeveloper.de/AllASCIITabellen.aspx

# ASCII-Codierung

Was für einem Zeichen entspricht

00110011? **3**

Welches Bit-Muster hat ein M?

**01001101**

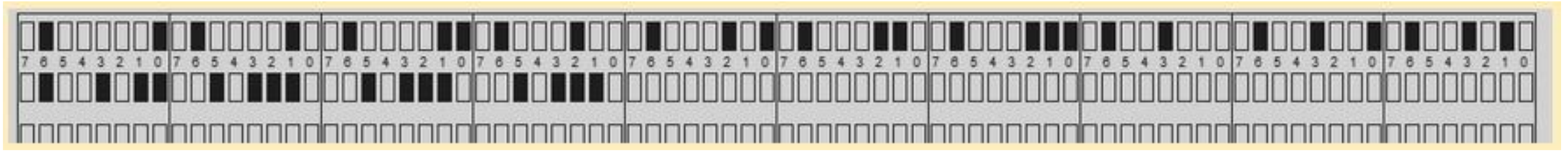
Welchem Zeichen entspricht

**$2D_{16}$**? -

| | | | | Bit 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Bit 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | Bit 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | Bit 4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** | **Hex** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| 0 | 0 | 0 | 0 | **0** | NUL | DLE | SP | 0 | @ | P | ` | p |
| 0 | 0 | 0 | 1 | **1** | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0 | 0 | 1 | 0 | **2** | STX | DC2 | " | 2 | B | R | b | r |
| 0 | 0 | 1 | 1 | **3** | EXT | DC3 | # | 3 | C | S | c | s |
| 0 | 1 | 0 | 0 | **4** | EOT | DC4 | $ | 4 | D | T | d | t |
| 0 | 1 | 0 | 1 | **5** | ENQ | NAK | % | 5 | E | U | e | u |
| 0 | 1 | 1 | 0 | **6** | ACK | SYN | & | 6 | F | V | f | v |
| 0 | 1 | 1 | 1 | **7** | BEL | ETB | ' | 7 | G | W | g | w |
| 1 | 0 | 0 | 0 | **8** | BS | CAN | ( | 8 | H | X | h | x |
| 1 | 0 | 0 | 1 | **9** | HAT | EM | ) | 9 | I | Y | i | y |
| 1 | 0 | 1 | 0 | **A** | IF | SUB | * | : | J | Z | j | z |
| 1 | 0 | 1 | 1 | **B** | VT | ESC | + | ; | K | [ | k | { |
| 1 | 1 | 0 | 0 | **C** | FF | FS | , | < | L | \ | l | \| |
| 1 | 1 | 0 | 1 | **D** | CR | GS | - | = | M | ] | m | } |
| 1 | 1 | 1 | 0 | **E** | SO | RS | . | > | N | ^ | n | ~ |
| 1 | 1 | 1 | 1 | **F** | SI | US | / | ? | O | _ | o | DEL |

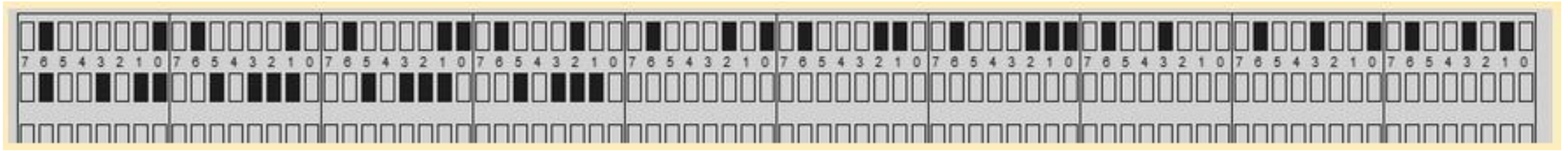
https://www.electronicdeveloper.de/AllASCIITabellen.aspx

# ASCII-Codierung: Lochkarte




Was ist auf der Lochkarte geschrieben? (Lochkarten werden gleich gelesen wie Deutscher Text! Von Oben-Links nach Rechts-Unten!)

# ASCII-Codierung: Lochkarte



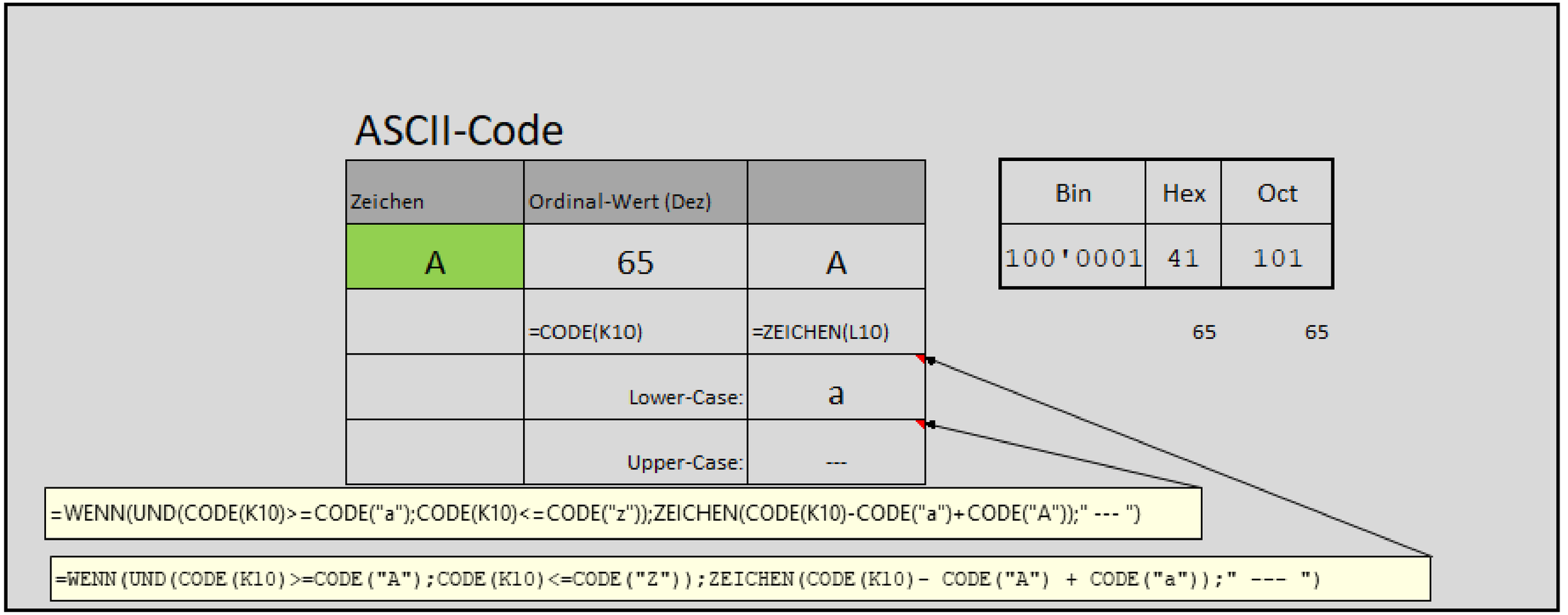

Was ist auf der Lochkarte geschrieben? (Lochkarten werden gleich gelesen wie Deutscher Text! Von Oben-Links nach Rechts-Unten!)

**ABCDEFGHIJ**

**KNNN**

# Aufgabe

Entwickeln Sie eine Excel-Tabelle für das Umwandeln von Klein- in Grossbuchstaben. Implementieren Sie dies über die Excel-Funktionen CODE() und ZEICHEN().



# Aufgabe

Entwickeln Sie eine Excel-Tabelle für das Umwandeln von einer HEX-Stelle in den Dezimalwert und umgekehrt. Implementieren Sie dies über die Excel-Funktionen CODE() und ZEICHEN().

| Umwandeln: | | | |
|---|---|---|---|
| Eine HEX-Stelle in Dezimal: | F | ==> | 15 |
| Dezimal in eine HEX-Stelle: | 14 | ==> | E |

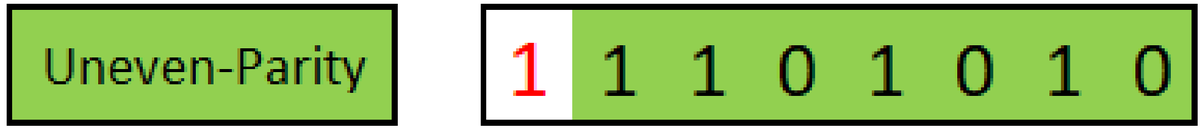
```
=WENN(UND(K23>="0";K23<="9");K23;
                              WENN(UND(K23>="A";K23<="F");CODE(K23)-CODE("A")+10;-1))

=WENN(UND(K24*1>=0;K24*1<=9);K24*1;
                              WENN(UND(K24*1>=10;K24*1<=15);ZEICHEN(K24+CODE("A")-10);" "))
```

# Was ist ein Paritätsbit?

**Gerade Parität (even parity) :** Die Anzahl der "1"-Bits im erweiterten Codewort ist gerade.


**Ungerade Parität (odd parity):** Die Anzahl der "1"-Bits im erweiterten Codewort ist ungerade.




Durch zufügen von **Redundanz** macht nicht jede 0/1 Kombination mehr Sinn und somit können gewisse Fehler erkannt werden!

Kann man gezielt mehr Redundanz zufügen und mehr Fehler erkannt und sogar korrigiert werde?

# Fehler korrigierender Code

Durch zufügen von **Redundanz** macht nicht jede 0/1 Kombination mehr Sinn und somit können gewisse Fehler erkannt werden!

Kann man gezielt mehr Redundanz zufügen und mehr Fehler erkannt und sogar korrigiert werde?

# Wie viele Bits sind nötig?

Wie viele Stellen (Bits) werden benötigt, um 256 verschiedene Binärcodes zu erzeugen?

$$2^x = 256 \quad ==> \quad x = \frac{\ln(256)}{\ln(2)}$$

Allgemeine Form:

$$\text{Min. Stellenzahl} = \text{Aufrunden}\left(\frac{\ln(\text{Anzahl Kombinationen})}{\ln(\textbf{Anzahl Zeichen pro Stelle})}\right)$$

Wie viele Stellen sind nötig:

| Bezeichnung | Zeichenvorrat / Stelle | Anzahl Zeichen / Stelle | Anzahl Kombinationen | | Anzahl Stellen | Anzahl Stellen (aufgerundet) |
|---|---|---|---|---|---|---|
| Dezimal | 0..9 | 10 | 1'000 | ==> | | |
| Oktal | 0..7 | | 4'096 | ==> | | |
| Hexadezimal | 0..9, A..F | | 100'000 | ==> | | |
| Alphabet | A..Z | | 20 | ==> | | |
| Binär | 0,1 | | 16 | ==> | | |
| BCD | 0,1 | | 10 | ==> | | |
| | Beliebig: | 16 | 1'000 | ==> | | |

# Wie viele Bits sind nötig? (Lösung)

Wie viele Stellen (Bits) werden benötigt, um 256 verschiedene Binärcodes zu erzeugen?

$$2^x = 256 \quad ==> \quad x = \frac{\ln(256)}{\ln(2)}$$

Allgemeine Form:

$$\text{Min. Stellenzahl} = \text{Aufrunden}\left(\frac{\ln(\text{Anzahl Kombinationen})}{\ln(\textbf{Anzahl Zeichen pro Stelle})}\right)$$

Wie viele Stellen sind nötig:

| Bezeichnung | Zeichenvorrat / Stelle | Anzahl Zeichen / Stelle | Anzahl Kombinationen | | Anzahl Stellen | Anzahl Stellen (aufgerundet) |
|---|---|---|---|---|---|---|
| Dezimal | 0..9 | 10 | 1'000 | ==> | 3.00 | 3 |
| Oktal | 0..7 | 8 | 4'096 | ==> | 4.00 | 4 |
| Hexadezimal | 0..9, A..F | 16 | 100'000 | ==> | 4.15 | 5 |
| Alphabet | A..Z | 26 | 20 | ==> | 0.92 | 1 |
| Binär | 0,1 | 2 | 16 | ==> | 4.00 | 4 |
| BCD | 0,1 | 2 | 10 | ==> | 3.32 | 4 |
| | Beliebig: | 16 | 1'000 | ==> | 2.49 | 3 |

# Redundanz

Der umgekehrte Weg ist ebenfalls häufig gefragt, wenn die Anzahl benötigten Bits verlangt sind, um z.B. bei einer A/D-Wandlung 1000 analoge Werte unterscheiden zu können. Man könnte nun in der obigen Auflistung  nachschauen, wieviel Bits nötig werden. Es geht aber auch hier mit einer einfachen Berechnung:

**«Anzahl Bit» = AUFRUNDEN von (LN(Anzahl Bitkombinationen) / LN(2))**

Zahlenbeispiel:
1000 Kombinationen sind verlangt. Die Berechnung dazu lautet...
«Anzahl Bit» = AUFRUNDEN von (LN(1000) / LN(2))
«Anzahl Bit» = AUFRUNDEN von (3 / 0.301)
«Anzahl Bit» = AUFRUNDEN von (9.966)
«Anzahl Bit» = 10
Kontrolle: 2 hoch 10 = 1024
Da aber «nur» 1000 Kombinationen verlangt sind, wurde um 24 Kombinationen über das Ziel hinausgeschossen. Die nächstkleiner Bitanzahl wäre 9. Dies würde allerdings nur 512 Kombinationen ergeben, was definitiv zu wenig wäre.

# Redundanz

In manchen Codes werden nicht alle möglichen Kombinationen gebraucht.
➔ Dieser Überschuss an Kombinationen wird Redundanz genannt

## Formel für die Berechnung

Redundanz = Anzahl verwendete Bits – Anzahl benötigte Bits

$$\text{Redundanz} = \text{Anzahl verwendete Bits} - \left(\frac{\ln(\text{Anzahl benötigter Kombinationen})}{\ln(\textbf{Anzahl Zeichen pro Stelle})}\right)$$

| Bezeichnung | Zeichenvorrat pro Stelle | Anzahl Zeichen / Stelle | Anzahl Kombinationen | Anzahl ver-wendete Stellen | | Berechnete Stellenzahl | min. Stellenzahl | Redundanz |
|---|---|---|---|---|---|---|---|---|
| Dezimal | 0..9 | 10 | 30 | 2 | ==> | 1.48 | 2 | 0.5229 |
| Oktal | 0..7 | 8 | 30 | 2 | ==> | 1.64 | 2 | 0.3644 |
| Hexadezimal | 0..9, A..F | 16 | 30 | 2 | ==> | 1.23 | 2 | 0.7733 |
| Alphabet | A..Z | 26 | 30 | 2 | ==> | 1.04 | 2 | 0.9561 |
| Binär | 0,1 | 2 | 30 | 5 | ==> | 4.91 | 5 | 0.0931 |
| BCD | 0,1 | 2 | 10 | 4 | ==> | 3.32 | 4 | 0.6781 |
| | Beliebig: | 16 | 1'024 | 3 | ==> | 2.50 | 3 | 0.5000 |

# Redundanz (Lösung)

In manchen Codes werden nicht alle möglichen Kombinationen gebraucht.
➔ Dieser Überschuss an Kombinationen wird Redundanz genannt

## Formel für die Berechnung

Redundanz = Anzahl verwendete Bits – Anzahl benötigte Bits

$$\text{Redundanz} = \text{Anzahl verwendete Bits} - \left(\frac{\ln(\text{Anzahl benötigter Kombinationen})}{\ln(\textbf{Anzahl Zeichen pro Stelle})}\right)$$

| Bezeichnung | Zeichenvorrat pro Stelle | Anzahl Zeichen / Stelle | Anzahl Kombinationen | Anzahl ver-wendete Stellen | | Berechnete Stellenzahl | min. Stellenzahl | Redundanz |
|---|---|---|---|---|---|---|---|---|
| Dezimal | 0..9 | 10 | 30 | 2 | ==> | 1.48 | 2 | 0.5229 |
| Oktal | 0..7 | 8 | 30 | 2 | ==> | 1.64 | 2 | 0.3644 |
| Hexadezimal | 0..9, A..F | 16 | 30 | 2 | ==> | 1.23 | 2 | 0.7733 |
| Alphabet | A..Z | 26 | 30 | 2 | ==> | 1.04 | 2 | 0.9561 |
| Binär | 0,1 | 2 | 30 | 5 | ==> | 4.91 | 5 | 0.0931 |
| BCD | 0,1 | 2 | 10 | 4 | ==> | 3.32 | 4 | 0.6781 |
| | Beliebig: | 16 | 1'024 | 3 | ==> | 2.50 | 3 | 0.5000 |

# Hammingdistanz

Die Hammingdistanz ist die Anzahl unterschiedlicher Bits zweier Codewörter.

Vergleich der Bitpositionen

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Codewort 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Codewort2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Vergleich | ✓ | ✓ | X | X | X | ✓ | ✓ | ✓ |

→Unterschied in 3 Bitpositionen

- Je höher die Hammingdistanz, desto besser wird die Fehlererkennung und

Fehlerbehebung von Codewörtern

# Hamming-Code -> Codieren

Wieviele Bits und mit welchen Werten muss ich Redundanz dazu fügen, dass Fehler Erkannt und sogar korrigiert werden können?

| F | 100 0110 |
|---|---|

Hamming-Code

| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

| | | | | |
|---|---|---|---|---|
| 12 | 1 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |

==>

| | | | |
|---|---|---|---|
| | | | |
| 1 | 0 | 1 | 1 |
| | | | |
| | | | |
| | | | |
| | | | |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| | | | |
| | | | |
| | | | |
| | | | |

| | | | | |
|---|---|---|---|---|
| Anzahl 1er: | 1 | 2 | 2 | 2 |
| Gerade/Ungerade: | Ungerade | Gerade | Gerade | Gerade |
| Even-Parity | 0 | 1 | 1 | 1 |

# Hamming-Code -> Decodieren (Fehler-Erkennung / Fehler Behebung)

Wieviele Bits und mit welchen Werten muss ich Redundanz dazu fügen, dass Fehler Erkannt und sogar korrigiert werden können?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Uebermittlungsfehler: | Nein | Nein | Ja | Nein | Nein | Nein | Nein | Nein | Nein | Nein | Nein | Nein |
| | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| Fehler erkannt: | | | | | | | | | | | | |

Mit Even-Parity übertragen!

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 12 |
| | | 1 | 0 | 1 | 1 | 11 |
| | | 1 | 0 | 1 | 0 | 10 |
| | | | | | | 9 |
| | | | | | | 8 |
| | | | | | | 7 |
| | | 0 | 1 | 1 | 0 | 6 |
| | | 0 | 1 | 0 | 1 | 5 |
| | | 0 | 1 | 0 | 0 | 4 |
| | | | | | | 3 |
| | | 0 | 0 | 1 | 0 | 2 |
| | | 0 | 0 | 0 | 1 | 1 |
| | Anzahl 1er: | 2 | 3 | 4 | 3 | |
| | Gerade/Ungerade: | Gerade | Ungerade | Gerade | Ungerade | |
| 10 | <== | 1 | 0 | 1 | 0 | |

# Vielen Dank für Ihre Aufmerksamkeit

**Walter Rothlin**
Bildungszentrum Uster